(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 022 860 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
09.04.2003 Bulletin 2003/15

(51) Int Cl.7: **H03M 13/29**

(21) Application number: **00101174.1**

(22) Date of filing: **21.01.2000**

(54) **Turbo decoding with variable number of iterations**

Turbo Dekodierung mit variabler Anzahl von Iterationen

Turbo décodage avec nombre d'itérations variable

(84) Designated Contracting States:
DE FR GB

(30) Priority: **22.01.1999 US 116765 P**
28.07.1999 US 363303

(43) Date of publication of application:
26.07.2000 Bulletin 2000/30

(73) Proprietor: **Sharp Kabushiki Kaisha**
Osaka-shi Osaka (JP)

(72) Inventor: **Srinivasa Somayazulu, V.**
OR 97223 (US)

(74) Representative: **Müller - Hoffmann & Partner Patentanwälte,**
Innere Wiener Strasse 17
81667 München (DE)

(56) References cited:
US-A- 5 761 248

- **BAUCH G ET AL: "ITERATIVE EQUALIZATION AND DECODING IN MOBILE COMMUNICATIONS SYSTEMS" PROCEEDINGS OF EUROPEAN PERSONAL AND MOBILE COMMUNICATIONS CONFERENCE, 30 September 1997 (1997-09-30), pages 308-312, XP002060630**
- **HAO R Y ; FOSSORIER M ; SHU LIN : "Two simple stopping criteria for iterative decoding " PROCEEDINGS 1998 IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY , 16 - 21 August 1998, page 279 XP002135987 CAMBRIDGE, MA, USA**
- **CHANG Y: "PARALLEL DECODING OF TURBO CODES" ELECTRONICS LETTERS,GB,IEE STEVENAGE, vol. 32, no. 13, 20 June 1996 (1996-06-20), pages 1188-1189, XP000599183 ISSN: 0013-5194**
- **CHANG Y: "Q-ary Turbo Codes with QAM Modulations" PROCEEDINGS OF ICUPC - 5TH INTERNATIONAL CONFERENCE ON UNIVERSAL PERSONAL COMMUNICATIONS, CAMBRIDGE, MA, USA , vol. 2, 29 September 1996 (1996-09-29) - 2 October 1996 (1996-10-02), pages 814-817, XP002135988**
- **SHAO R Y ET AL: "TWO SIMPLE STOPPING CRITERIA FOR TURBO DECODING" IEEE TRANSACTIONS ON COMMUNICATIONS,US,IEEE INC. NEW YORK, vol. 47, no. 8, August 1999 (1999-08), pages 1117-1120, XP000848102 ISSN: 0090-6778**

EP 1 022 860 B1

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to channel coding for digital communications systems and, more particularly, to a turbo code decoder useful for channel coding and a method for optimizing the performance of a turbo code decoder.

[0002] Forward error correction (FEC) is a system of error control for data transmission systems where the receiver is capable of detecting and correcting errors in the "as received" message induced by noise in the transmission channel. FEC is useful in connection with data transmission systems which lack a reverse channel with which retransmission of data can be requested or where retransmission would be difficult because the delay would be excessive or repeated retransmission would be required because of the number of expected errors. For these reasons, FEC has been of particular interest and use in wireless communication and space probe and satellite data transmission systems. FEC relies on channel coding where input message sequences are mapped to code symbol sequences that add redundancy and memory to the data before transmission.

[0003] Generally, channel coding utilizes block or convolutional coding to add redundancy to the message bit stream. Block coding breaks the bit stream representing the message into fixed size blocks and independently adds redundant code symbols to each block. Block coding is usually decoded with algebraic techniques. On the other hand, convolutional coding continuously adds redundant symbols to the bit stream based on the contents of the stream. In the convolutional encoder, the bits of the message are shifted serially into and out of a shift register having a number of individual registers. The output code is the result of modulo arithmetic performed on the contents of the shift register and, in some cases, the input bit stream as each successive message symbol or bit is shifted into the register. While bit stream segmentation is not required for convolutional coding, the coded bit stream is typically broken into blocks or frames for other reasons before transmission. Decoding of convolutional codes is accomplished with a heuristic or trial-and-error approach.

[0004] Turbo codes are produced by encoders comprising two, or more, parallel, constituent encoders. The constituent encoders are often, but not necessarily, identical convolutional encoders. An interleaver or permuter is attached to the input of one or more of the constituent encoders. The interleaver rearranges the input of the attached constituent encoder in a systematic, pseudo-random-manner. As a result, turbo codes comprise two or more (depending on the number of encoders) independently coded symbol streams that refer to the same input information bit stream. Turbo codes are of particular interest because with a relatively simple constituent code and large interleaver their performance can be near the theoretical or Shannon limit of the transmission channel.

[0005] Turbo code decoding is an iterative process with the results of a first modular decoder forming part of the input to a second modular decoder and so forth until the required number of iterations is achieved. When the turbo code is composed of two parallel concatenated codes, the modular turbo code decoder comprises two serially connected constituent decoders separated by an interleaver that reorders the output of the first decoder so that it may be used as input to the next decoder. Decoders for turbo codes with more than two parallel constituent codes may take a number of forms. A convolutional encoder is a state machine that codes by tracing a path through a code tree or trellis on the basis of the sequence of input symbols. From the symbols of the "as received," coded message the convolutional code decoder attempts to retrace the encoder's path through the code tree or trellis outputting the symbols of the decoded message while correcting errors incurred in transmission. One technique for decoding convolutional codes relies on algorithms which retrace the path of "maximum likelihood" through the trellis. One such "maximum likelihood" algorithm used in turbo code decoding is the soft output Viterbi algorithm (SOVA). A constituent decoder applying the SOVA algorithm computes or estimates the "log likelihood ratio", the logarithm of the ratio of the conditional probabilities of receiving the two outcomes (binary "1" and "0") given the observed signal value. The output of the constituent decoder is a plurality of signed numbers. The sign expresses the plurality of the decoded message symbol. The magnitude is a "soft" or analog value expressing the probability that the decoded symbol is the same as the original message symbol.

[0006] Bauch, Khorram, and Hagenauer, "Iterative equalisation and decoding in mobile communications systems", Proceedings of European Personal And Mobile Communication Conference, 30 September 1997, pages 308-312, discloses usual iteration abortion criteria for turbo decoders. Additionally to the well-known cross entropy criterion the "hard decision" criterion is described, in which the convergence of the iterations is supposed to be achieved when the hard decisions stop changing, and the "risk function" criterion is described, which is based on an observation of the distribution of the Log-likelihood values in a block of data, basically by comparing to a threshold an average of the likelihood of wrong decisions in a data block. Document Hao, Fossorier and Shu, "Two simple Stopping Criteria for Iterative Decoding", Proceedings IEEE 1998, International Symposium On Information Theory, 16 to 21 August 1998, page 279, mentions the "hard decision" criterion and a further criterion based on the observation of the number of changes of signs of the Log-likelihood ratios over the iterations.

[0007] Generally, the turbo code decoder converges on a final decoded symbol sequence with successive iterations

and the error rate performance improves until a threshold number of iterations is reached. While the error rate performance of the decoder generally improves with additional iterations, the rate of improvement decreases. Each iteration takes time further delaying completion of decoding. Heretofore, the number of iterations to be performed by a particular turbo code decoder was hard wired into the decoder. Optimizing the number of iterations to be hardwired into the decoder involves compromises in the error rate and latency of the decoder's performance. Further, due to the random nature of noise, "as received" data sequences are unequally corrupted and require different numbers of iterations to achieve the same level of error correction.

[0008] What is desired, therefore, is a turbo code decoder and a method of decoding that optimize the performance of the decoder producing a given level of error correction in the fewest number of decoding iterations on the average. Further, it is desired that the operation of the decoder be responsive to the error correcting requirements of each message. Optimizing the decoding process reduces the latency in decoding a message at an acceptable error rate and reduces the cost and complexity of the decoder.

SUMMARY OF THE INVENTION

[0009] The present invention overcomes the aforementioned drawbacks of the prior art by providing a method of optimizing the performance of an iterating turbo code decoder including at least one constitutent decoder, comprising the steps of: (a) establishing a limit for outputs of at least one constituent decoder, respectively; (b) determining a number of said outputs approximately equaling said limit for each iteration by said at least one turbocode decoder; wherein (c) a first number of an output approximately equaling said limit produced by a first serial constituent decoder performing a second iteration is determined; (d) a second number of an output approximately equaling said limit produced by a last said serial constituent decoder performing a second iteration is determined; and (e) the operation of said turbocode decoder is terminated when said first number and said second number are substantially equal.

[0010] In a preferred embodiment, a third number of an output approximately equaling said limit produced by said last serial constituent decoder while performing a first iteration is determined, and the operation of the turbocode decoder is terminated when said first, second and third numbers are substantially equal.

[0011] The progress of a turbo code decoder in decoding a message sequence can be monitored by establishing a limit for the output of a constituent decoder and monitoring the number of outputs of equaling or approaching the limit. If the number of outputs approaching the limit or saturating does not change for sucessive iterations by the turbo code decoder, no progress is being made in decoding the message. Operation of the turbo code decoder can be terminated without loss of data.

[0012] In other words: A technique for applying the method of optimizing the performance of an iterating turbo code decoder including a plurality of serial constituent decod-ers comprises establishing a limit for an output of a constituent decoder; determining a first number of outputs produced by the first serial constituent decoder performing an iteration approximately equaling the limit; determining a second number of outputs produced by the last serial constituent decoder performing the iteration approximately equaling the limit; and terminating operation of the turbo code decoder when the first and second numbers of outputs are substantially equal.

[0013] By monitoring the progress of the decoding process and terminating decoding iterations when progress slows or stops, the delay in decoding any particular message stream can be minimized while achieving a satisfactory error rate for any "as received" symbol sequence.

[0014] An optimized iterating turbo code decoder is also provided which comprises at least one constituent decoder, a comparator to compare outputs of said constituent decoders to a threshold value assigned to said outputs, respectively, which further shows a plurality of serial constituent decoders, a counter to count at least a first and a second number of outputs produced, respectively, by a first said serial constituent decoder and a subsequent said serial constituent decoder approximately equaling said threshold value, and a decision unit to terminate the operation of said turbo code decoder when said first number is approximately equal to said second number.

[0015] In a second embodiment of the optimized turbo code decoder, the number of outputs of the first and last of the serial constituent decoders that have saturated during an iteration is compared to determine progress in decoding. In a third embodiment, the number of outputs of the first serial constituent decoder performing a first iteration and the numbers of outputs of the last serial constituent decoder performing the first and a subsequent iteration are compared to determine decoding progress and whether further iterations are warranted.

[0016] The foregoing and other objectives, features, and advantages of the invention will be more readily understood upon consideration of the following detailed description of the invention, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017]

FIG. 1 is a block diagram of a communication link incorporating channel coding.
FIG. 2 is a block diagram of an exemplary systematic, recursive convolutional encoder.
FIG. 3 is a trellis diagram of the operation of a recursive, constituent convolutional encoder of the encoder of FIG. 2.
FIG.4 is a block diagram of a turbo code decoder according to the present invention.
FIG. 5 is trellis diagram for a soft output Viterbi algorithm based, constituent, convolutional decoder.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0018] Referring to FIG. 1, forward error correction (FEC) is used to detect and correct errors which can occur in digital messages as a result of noise in a communication link 2. A message 4 comprising a sequence of bits or symbols (m_1, m_2, ... m_i, ...) originates with an information source 6. If the communication link 2 incorporates FEC, the message sequence 4 is encoded before transmission to a code symbol sequence 8, (C_1, C_2, ... C_i, ...) by an encoder 10. The codeword sequence 8 is forwarded to a modulator 12 where it is converted to signals 14 ($\{s_i(t)\}$) which are suitable for the transmission in the channel 16 of the communication link 2. The transmission channel 16 may be affected by noise resulting in a distorted signal 18 ($\{\hat{s}_i(t)\}$) at the receiver 20. In the receiver 20, a demodulator 22 converts the signal with any distortion incurred during transmission into a demodulated bit stream 24 (Z_1, Z_2, ... Z_i, ...). The demodulated bit stream 24 is then decoded by a decoder 26 which identifies and corrects errors in the "as received" message to produce a corrected, decoded output message stream 28 (m'_1, m'_2, ... m'_i, ...). The output message is passed to an information sink 30 for consumption.

[0019] Several channel coding techniques are used for FEC in communication systems. One technique, which is thought to achieve performance near the theoretical or Shannon limit of the transmission channel, is turbo coding. Turbo codes are interleaved, parallel concatenated, codes. The constituent codes are often convolutional codes but may be codes produced by other techniques.

[0020] Referring to FIG. 2, an exemplary turbo-code encoder 40 (indicated by a bracket) comprises two parallel, recursive, systematic, convolutional, constituent encoders 42 and 44 (indicated by brackets) with an interleaver 46 attached to the input of one of the encoders 44. Turbo code encoders may have more than two parallel constituent encoders with additional interleavers attached to the inputs of the additional constituent encoders. The turbo code encoder 40 has three intermediate outputs (c_0) 47, (c_{C1}) 48, and (c_{C2}) 50 which are combined into a coded serial bit stream (C) 52 by an output multiplexer 54. The code rate of the turbo code encoder 40 or the ratio of the number of input message symbols (m) to the number of output code symbols (C) is 1/3.

[0021] The turbo code encoder 40 is a systematic encoder with the symbols of the input message or frame (m) 56 constituting one of encoder's intermediate outputs (c_0) 47. The intermediate outputs (c_{C1}) 48 and (c_{C2}) 50 are the outputs of the first 42 and second 44 constituent convolutional encoders, respectively. The first 42 and second 44 constituent convolutional encoders each comprise a shift register 58 and a pair of modulo-2 adders 60 and 62. The shift registers 58 of the illustrated encoders have two individual registers; (T1) 64 and (T2) 66. The symbols of an input bit stream for the constituent encoders 42 and 44, (m_i) 56 or ($m(\alpha)_i$) 68, respectively, are shifted one bit at a time into the shift register 58. To ensure that the contents of the shift registers 58 are known at the start of the message and that all of the message bits are shifted completely through each shift register 58, the turbo code encoder includes a padding unit 70 to add sufficient bits (usually zeroes) to "flush" the shift register to a known state at the end of the message or frame. A convolutional encoder adds redundant bits to the message on a continuous basis so that the entire message is one code. However, the coded message data is often segmented into fixed length frames for transmission.

[0022] In the turbo code encoder 40 the input of the first constituent convolutional encoder 42 is the exemplary message sequence (m) 56 [10101]. In the recursive convolutional encoders 42 and 44, the modulo-2 sum of the input bit and the contents of the two registers (T1) 64 and (T2) 66 is computed in a modulo-2 adder 60 to create a feedback bit 72 (F). The output of the constituent convolutional encoder, (c_{C1}) 48 or (c_{C2}) 50, is the modulo-2 sum of the feedback bit 72 (F) and the contents of the T2 register 66 computed in the adder 62. The output of the first constituent convolutional encoder (c_{c1}) 42 is the coded bit stream: [11011].

[0023] The input ($m(\alpha)$) 68 of the second constituent convolutional encoder 44 is the message sequence (m) 56 as rearranged by an interleaver 46. Interleaving rearranges the input symbols in some pseudo-random manner so that the multiple code symbols representing each input symbol will be separated from each other in the output bit stream (C) 52. The error correcting performance turbo codes is due, in part, to the use of large turbo interleavers that interleave hundreds bits. Interleaving results in two or more (depending on the number of constituent codes) independently coded symbol streams that refer to the same message bit stream. Turbo code decoders exploit this structure of the code to

derive extrinsic information about bits in one symbol stream from the bits in the remaining stream(s) and use this information to estimate the correctly decoded symbols. The estimates are refined during successive iterations of the decoding process.

[0024] In the exemplary interleaver 46 the input symbols are rearranged according to a table where the first bit of the input sequence (m_1) becomes the fourth bit ($m(\alpha)_4$) of the convolutional encoder input sequence 68; the second input bit (m_2) becomes the third bit ($m(\alpha)_3$) and so forth. As a result, the output (c_{C2}) 50 of the second convolutional encoder 44 is different from that of the first constituent encoder 42 even though the encoding process within the encoder is the same. Further, as a result of interleaving, the three coded output symbols representing any input message symbol are separated in the output bit stream 52.

[0025] A convolutional encoder is a state machine and its operation can be represented by a path traced through a code tree or a trellis diagram. FIG. 3 is a trellis diagram illustrating an exemplary operation of the constituent recursive convolutional encoder 42 of FIG. 2. Each node 80 of the trellis corresponds to a state of the shift register of the encoder 42 (listed vertically to the left of the trellis) at an input time or coding stage 82 (t_0, t_1,...) indicated horizontally across the top of the trellis). Two paths exit each node of the trellis. The path to be followed to the next state at the next stage of encoding is determined by the message bit causing the state transition. An input message bit (1) causes the encoder to follow the path represented by a dashed line and an input message bit (0) causes the encoder to follow the path represented by a solid line. An output code symbol 84 for each path is shown adjacent to the path. For example, at t_1 a bit (1) is shifted into the encoder 42 causing the encoder to transition from the an intial state (00) 86 (as a result of padding) to the next state (10) 88. When the encoder exits the first node (00) 86 for the next state (10) 88, it exits on the path corresponding to the input bit (1) (dashed line) which causes the encoder to output a code symbol (11) 90 Likewise, the next message bit (0) determines the exit path from the second node (state 10 at t_1) 88 producing the next code symbol (01) 92 and causing the encoder to move to its next state (11 at t_2) 94 . It is desirable for the encoder to start and end in an all zero state. For an input sequence [101] two extra bits [01] are necessary to produce a zero ending state. Following the path through the trellis for an input message sequence [101] plus two padding bits [01], produces the output code: [1101100111]. The output of the encoder of FIG. 2 is the output constituent encoders (the trellis diagrams for each of the two encoders) which is multiplexed with the original message symbols.

[0026] Turbo code decoding is an iterative process and the turbo code decoder is modular in nature with the output of a first module (the result of a first iteration) being the part of the input to a first decoder of the next modular decoder for use in the next iteration of the process. Referring to FIG. 4, an exemplary modular turbo code decoder includes a number of serially connected, constituent decoders 100 and 102 equal to the number constituent encoders in the particular turbo code encoder. Other arrangements of the decoder are possible if the number of constituent codes is greater than two, but the present invention may be utilized with these other arrangements, as well. The "as received" coded transmission (Z) 104, including any errors incurred in transmission, is received by the decoder from the demodulator 22. In an input demultiplexer 106, the "as received" data stream (Z) 104 is separated into three constituent streams; (z_0) 110, (z_{C1}) 108, and (z_{C2}) 112 representing the "as received" versions of the three intermediate outputs of the encoder; (c_0) 47, (c_{C1}) 48, and (c_{C2}) 50.

[0027] In the constituent decoders 100 and 102, the soft output Viterbi algorithm (SOVA) is used to retrace the "most likely" path followed by the encoder through the trellis diagram when it produced the encoded message. Referring to FIG.5, in applying the Viterbi algorithm the decoder operates serially through the stages. At each stage, a path metric 170 is computed for paths from the initial state to all possible states for that stage. The path metric 170 is the accumulated difference metric 172 along each path from the decoder's initial state 174 to each node 176 at the particular stage. The difference metric 172 for a path between two nodes represents a measure of the difference between a code word of the received message and the corresponding code word which the encoder would have generated in making the transition between those two nodes or states. According to the Viterbi algorithm, the "most likely" path followed by the encoder in coding the sequence is the path with the least path metric. This path is retained for use at the next input stage and the path with the greater path metric is discarded. As the algorithm is applied for input stages further and further into the trellis, a single "most likely" path emerges after a delay 180. There is a high probability that errors in the received message will be eliminated and that the message will be correctly decoded by following this path through the trellis diagram.

[0028] A modified Viterbi algorithm (the soft output Viterbi algorithm or SOVA) is used in turbo code decoders to produce "soft" or analog outputs from soft or hard inputs. "Soft" inputs and outputs quantize the level of the actual signal (often at eight levels) between the possible a binary values of the signal. The "soft" value represents the reliability of the signal's value as well as its polarity.

[0029] There are two basic steps in the SOVA algorithm used in the constituent decoders of the turbo code decoder. The first or Viterbi step is similar to the Viterbi algorithm with the addition of the computation of the maximum path metric differences for each node at each stage of the trellis. In the second or update step, a window 178 is established corresponding to the delay of the Viterbi algorithm in converging on a single path. The window 178 slides along the trellis as the decoder moves from stage to stage. The minimum path metric difference is found from among all possible

paths in this window that compete with the survivor path and would have led to a different bit decision than the survivor path. This minimum path metric difference is given a sign based on the hard value of the current bit and is the soft value of the SOVA output. The output of a constituent convolutional decoder applying the SOVA algorithm comprises a signed number for each decoded bit of the message. The sign of the number (the sign of the "log likelihood ratio") represents the value or polarity of the bit. The magnitude of the number represents the reliability of its proper decoding (the magnitude of the "log likelihood ratio").

[0030] With the SOVA algorithm, the path metric computations of the Viterbi algorithm are augmented by the addition of a term which incorporates extrinsic information (Lin(t)) supplied by the previous constituent decoder. For each state (k) at a particular t in the trellis, the path metric $M_{k,t}$ is computed by extending the path metrics from all states (k'), at time t-1, with valid transitions to a state k, at time t, as follows:

$$M_{k,t} = \max_{k'} \left\{ M_{k',t-1} + \tfrac{1}{2} \sum_{v=1}^{N} Lc\, y_{v,t}\, x_{v,k',k} + \tfrac{1}{2} L_{in}(t) \right\}$$

where:

Lin(t) is the extrinsic information from the previous decoder, and
Lc is a channel reliability factor equal to:

$$Lc = 4\, a\, Es/N_0$$

where:

a = the fading amplitude of the channel; and
Es/N_0 = signal to noise ratio for the channel.

[0031] Referring to FIG. 4, the first constituent decoder 100 applies the SOVA algorithm to the systematic data (Z_0) 110, the coded data from the first encoder (Z_{C1}) 108, and initial a-priori information ($L_{in1,i}$) 114 to compute a soft output (L(sova1) 116 comprising a plurality of signed numbers each representing the value of a decoded message bit and the probability that the value has been correctly decoded. The extrinsic information (L(e)) 120 is produced by subtracting the product of systematic data (Z_0) 110 and the channel reliability factor (Lc) 124 produced in the multiplier 122 and the a-priori information ($L_{in1,i}$) 114 from the decoder output (L(sova)) 116 in a subtraction unit 118. After interleaving the extrinsic information (L(e))120 becomes a-priori information (Lin2,j) 126-which is supplied to the second constituent decoder 102.

[0032] The input data for the second constituent decoder 102 comprises the interleaved version of a-priori information (Lin2,2j) 126 from the first constituent decoder 100, the systematic data (Z_0) 110 which has also been interleaved in the interleaver 128, and the "as received" coded output of the second encoder of the turbo code encoder (Z_{C2}) 112 which has been separated from the input message bit stream, (Z) 104 by the input demultiplexer 106. The second constituent decoder 102 applies the SOVA algorithm to this second coded information and produces a second soft output (Lsova2) 130, completing an iteration of the decoding process.

[0033] If the required number of iterations have been completed, the soft output of the second decoder (Lsova2) 130 is diverted by switch 132 to the deinterleaver 134 and passed to an conversion unit 136 where "hard" (binary) values are selected for the decoded message 138 from the "soft" (analog) values of the decoder output.

[0034] If the required number of iterations has not been completed, the interleaved intrinsic information (Lin2,j) and the interleaved systematic data (Z_0) 110 from the input to the second decoder 102 is subtracted from the soft output of the second decoder (Lsova2) 130 in the subtraction unit 140. The resulting extrinsic information (Le) 142 is an interleaved version of the intrinsic information (Lin1,j+1) 144. After deinterleaving in the deinterleaver 134, this intrinsic information (Lin1,j+1) 144 is fed back through the switch 146 to the input to the first constituent decoder 100 for the next successive iteration of the decoding process.

[0035] The output of the SOVA algorithm depends upon the value of the minimum of the maximum path metric differences in the update window along the survivor path. For low signal to noise ratios (SNR) the maximum path metric differences are likely to be small. For high SNR, the path metric differences are likely to be large resulting in large values of the SOVA output (Lsova). These large values have a negative effect on the convergence of the algorithm

and the performance of the algorithm can be improved by limiting the output (Lsova).

[0036] The present inventor came to unexpected realization that the convergence of the SOVA algorithm can be monitored by comparing the number of outputs of the constituent decoders that are saturating at a limiting level at each iteration and discontinuing iteration when this number does not change from iteration to iteration. Further, the decision to continue iterating can be made by comparing the change in the number of saturating outputs produced by each of the individual constituent decoders in a single iteration.

[0037] In the present invention, the outputs (L(sova)) are directed from the constituent decoders 100 and 102 to a comparator 150 which compares each output to a threshold or limit (T). The numbers of L(sova) outputs from the second constituent decoder 102 (L(sova2)) for a first iteration (N2,j) and a successive iteration (N2, j+1) and from the first constituent decoder 100 for the successive iteration (N1,,j+1) equaling the threshold (T) are counted by the counter 151 and stored in memory 152, 154, and 156. The three numbers of saturated L(sova) values are compared in a decision unit 158. If the three numbers are approximately equal, the algorithm can be declared to have converged and the operation of the turbo code decoder can be terminated 162 without loss of information. If the three numbers are not equal, the algorithm is continuing to converge and the turbo code decoder will be allowed to continue 164 to operate unless the maximum number of iterations 160 has been reached. A second technique and apparatus stores and compares the numbers of outputs of the first and second constituent decoders which have saturated in successive iterations. When the numbers of saturating outputs do not in succeeding iteration does not change convergence is declared and operation of the decoder is halted. Likewise, the saturating outputs of a single constituent decoder can monitored for successive iterations until there is little or no further change in the numbers of saturating outputs. A fourth, less complex technique is to compare the output of the last serial constituent decoder with the output of the first serial constituent decoder performing one iteration. While the convergence of the algorithm is indicated by a lack of change in the number of saturated outputs produced by the first and last decoders performing a single iteration, there is less certainty of the effect of continuing operation using this method. If all of the values in the SOVA outputs of the constituent decoders have saturated the algorithm has converged and operation may be terminated. The threshold (T) and the L(sova) value are both signed values. The number of comparisons may be reduced by 50% by comparing outputs of only one sign with threshold having the same sign.

Claims

1. A method for optimizing the performance of an iterating turbocode decoder including at least one constituent decoder (100, 102), comprising the steps of:

   (a) establishing a limit for outputs of at least one constituent decoder (100, 102), respectively;
   (b) determining a number of said outputs approximately equaling said limit for each iteration by said at least one turbocode decoder;
   wherein
   (c) a first number of an output approximately equaling said limit produced by a first serial constituent decoder (100) performing a second iteration is determined;
   (d) a second number of an output approximately equaling said limit produced by a last said serial constituent decoder (102) performing a second iteration is determined; and
   (e) the operation of said turbocode decoder is terminated when said first number and said second number are substantially equal.

2. The method of claim 1, wherein a third number of an output approximately equaling said limit produced by said last serial constituent decoder (102) while performing a first iteration is determined; and
   the operation of said turbocode decoder is terminated when said first, second and third number are substantially equal.

3. The method of claim 1 or claim 2, wherein said first, second and third number of said outputs are numbers of said outputs of one sign approximately equaling said limit having said same sign.

4. The method of claims 1 or 2, wherein at least one of said constituent decoders (100, 102) applies the soft output Viterbi algorithm.

5. An iterating turbocode decoder, comprising:

   (a) at least one constituent decoder (100, 102),

(b) a comparator (150) to compare outputs of said constituent decoders (100, 102) to a threshold value assigned to said outputs, respecticely,
characterized by
(c) a plurality of serial constituent decoders,
(d) a counter (151) to count at least a first and a second number of outputs produced, respectively, by a first said serial constituent decoder (100) and a subsequent said serial constituent decoder (102) approximately equaling said threshold value; and
(e) a decision unit (158) to terminate the operation of said turbocode decoder when said first number is approximately equal to said second number.

6. The turbocode decoder of claim 5, **characterized by** a memory (152, 154, 156) to store the first, the second and a third number of outputs approximately equaling said threshold value, said outputs produced, respectively, by the subsequent said serial constituent decoder (102) performing a first and a subsequent iteration, and the first said serial constituent decoder (100) performing said subsequent iteration, wherein the decision unit (158) terminates the operation of said turbocode decoder when said first, second and third number are approximately equal.

7. The turbocode decoder of claims 5 or 6, wherein said constituent decoder (100, 102) is a soft output Viterbi algorithm decoder.

Patentansprüche

1. Verfahren zum Optimieren des Funktionsvermögens eines iterierenden Turbocode-Decodierers mit mindestens einer Decodiererkomponente (100, 102), mit den folgenden Schritten:

(a) Erstellen eines jeweiligen Grenzwerts für Ausgangswerte mindestens einer Decodiererkomponente (100, 102);
(b) Bestimmen einer Anzahl von Ausgangswerten, die näherungsweise dem Grenzwert entsprechen, für jede Iteration durch den mindestens einen Turbocode-Decodierer;
- wobei
(c) eine erste Anzahl von Ausgangswerten, die näherungsweise dem Grenzwert entsprechen und durch die erste serielle Decodiererkomponente (100) beim Ausführen einer zweiten Iteration erzeugt wurden, bestimmt wird;
(d) eine zweite Anzahl von Ausgangswerten, die näherungsweise dem Grenzwert entsprechen und durch die letzte serielle Decodiererkomponente (102) beim Ausführen einer zweiten Iteration erzeugt wurden, bestimmt wird; und
(e) der Betrieb des Turbocode-Decodierers beendet wird, wenn die erste und die zweite Anzahl im Wesentlichen gleich sind.

2. Verfahren nach Anspruch 1, bei dem

- eine dritte Anzahl von Ausgangswerten, die näherungsweise dem genannten Grenzwert entsprechen und die von der letzten seriellen Decodiererkomponente (102) beim Ausführen einer ersten Iteration erzeugt wurden, bestimmt wird; und
- der Betrieb des Turbocode-Decodierers beendet wird, wenn die erste, die zweite und die dritte Anzahl im Wesentlichen gleich sind.

3. Verfahren nach Anspruch 1 oder 2, bei dem die erste, die zweite und die dritte Anzahl der Ausgangswerte Zahlen der Ausgangswerte mit einem Vorzeichen sind, die näherungsweise dem Grenzwert mit demselben Vorzeichen entsprechen.

4. Verfahren nach einem der Ansprüche 1 oder 2, bei dem mindestens eine der Decodiererkomponente (100, 102) den Viterbialgorithmus mit weichen Ausgangswerten anwendet.

5. Iterierender Turbocode-Decodierer mit:

(a) mindestens einer Decodiererkomponente (100, 102);
(b) einem Komparator (150) zum Vergleichen von Ausgangswerten der Decodiererkomponenten (100, 102)

mit einem jeweiligen Schwellenwert, wie er jedem der Ausgangswerte zugewiesen ist;
gekennzeichnet durch
(c) mehrere serielle Decodiererkomponenten;
(d) einen Zähler (151) zum jeweiligen Zählen zumindest einer ersten und einer zweiten Anzahl von Ausgangswerten, wie sie von einer ersten seriellen Decodiererkomponente (100) bzw. einer folgenden seriellen Decodiererkomponente (102) erzeugt werden, und die näherungsweise dem Schwellenwert entsprechen; und
(e) eine Entscheidungseinheit (158) zum Beenden des Betriebs des Turbocode-Decodierers, wenn die erste Anzahl näherungsweise der zweiten Anzahl entspricht.

6. Turbocode-Decodierer nach Anspruch 5, **gekennzeichnet durch** einen Speicher (152, 154, 156) zum Speichern der ersten, der zweiten und der dritten Anzahl von Ausgangswerten, die näherungsweise dem Schwellenwert entsprechen, wobei diese Ausgangswerte **durch** die folgende serielle Decodiererkomponente (102) beim Ausführen einer ersten und einer folgenden Iteration bzw. die erste serielle Decodiererkomponente (100) beim Ausführen der folgenden Iteration erzeugt wurden, wobei die Entscheidungseinheit (158) den Betrieb des Turbocode-Decodierers beendet, wenn die erste, die zweite und die dritte Anzahl näherungsweise gleich sind.

7. Turbocode-Decodierer nach einem der Ansprüche 5 oder 6, bei dem die Decodiererkomponente (100, 102) ein Decodierer mit einem Viterbialgorithmus mit weichen Ausgangswerten ist.

Revendications

1. Procédé pour optimiser la performance d'un décodeur de turbo code du type à itérations comprenant au moins un décodeur constitutif (100, 102), le procédé comprenant les étapes consistant à :

(a) établir une limite pour les sorties d'au moins un décodeur constitutif (100, 102), respectivement ;
(b) déterminer un nombre desdites sorties valant approximativement ladite limite pour chaque itération par ledit au moins un décodeur de turbo code ;
dans lequel
(c) un premier nombre d'une sortie valant approximativement ladite limite qui est produite par un premier décodeur constitutif sériel (100) effectuant une seconde itération est déterminé ;
(d) un second nombre d'une sortie valant approximativement ladite limite qui est produite par un dernier dit décodeur constitutif sériel (102) effectuant une seconde itération est déterminé ; et
(e) il est mis fin aux opérations dudit décodeur de turbo code lorsque ledit premier nombre et ledit second nombre sont sensiblement égaux.

2. Procédé selon la revendication 1, dans lequel un troisième nombre d'une sortie valant approximativement ladite limite qui est produite par ledit dernier décodeur constitutif sériel (102) pendant l'exécution d'une première itération est déterminé ; et
il est mis fin au fonctionnement dudit décodeur de turbo code lorsque lesdits premier, second et troisième nombres sont sensiblement égaux.

3. Procédé selon la revendication 1 ou la revendication 2, dans lequel lesdits premier, second et troisième nombres desdites sorties sont des nombres desdites sorties d'un certain signe valant approximativement ladite limite ayant ledit même signe.

4. Procédé selon la revendication 1 ou la revendication 2, dans lequel au moins l'un desdits décodeurs constitutifs (100, 102) applique l'algorithme de Viterbi à sortie souple.

5. Décodeur de turbo code du type à itérations, comprenant :

(a) au moins un décodeur constitutif (100, 102),
(b) un comparateur (150) pour comparer les sorties desdits décodeurs constitutifs (100, 102) à une valeur de seuil affectée auxdites sorties, respectivement,
caractérisé par
(c) une multiplicité de décodeurs constitutifs sériels,
(d) un compteur (151) pour compter au moins un premier nombre et un second nombre de sorties produites, respectivement, par un premier dit décodeur constitutif sériel (100) et un dit décodeur constitutif sériel subsé-

quent (102) et valant approximativement ladite valeur de seuil ; et
(e) une unité de décision (158) pour mettre fin aux opérations dudit décodeur de turbo code lorsque ledit premier nombre est approximativement égal audit second nombre.

6. Décodeur de turbo code selon la revendication 5, **caractérisé par** une mémoire (152, 154, 156) pour stocker le premier nombre, le second nombre et un troisième nombre de sorties valant approximativement ladite valeur de seuil, lesdites sorties étant produites, respectivement, par ledit décodeur constitutif sériel subséquent (102) exécutant une première itération et une itération subséquente, et par le premier dit décodeur constitutif sériel (100) exécutant ladite itération subséquente, dans lequel l'unité de décision (158) met fin aux opérations dudit décodeur de turbo code lorsque lesdits premier, second et troisième nombres sont approximativement égaux.

7. Décodeur de turbo code selon la revendication 5 ou la revendication 6, dans lequel ledit décodeur constitutif (100, 102) est un décodeur appliquant l'algorithme de Viterbi à sortie souple.

FIG.1

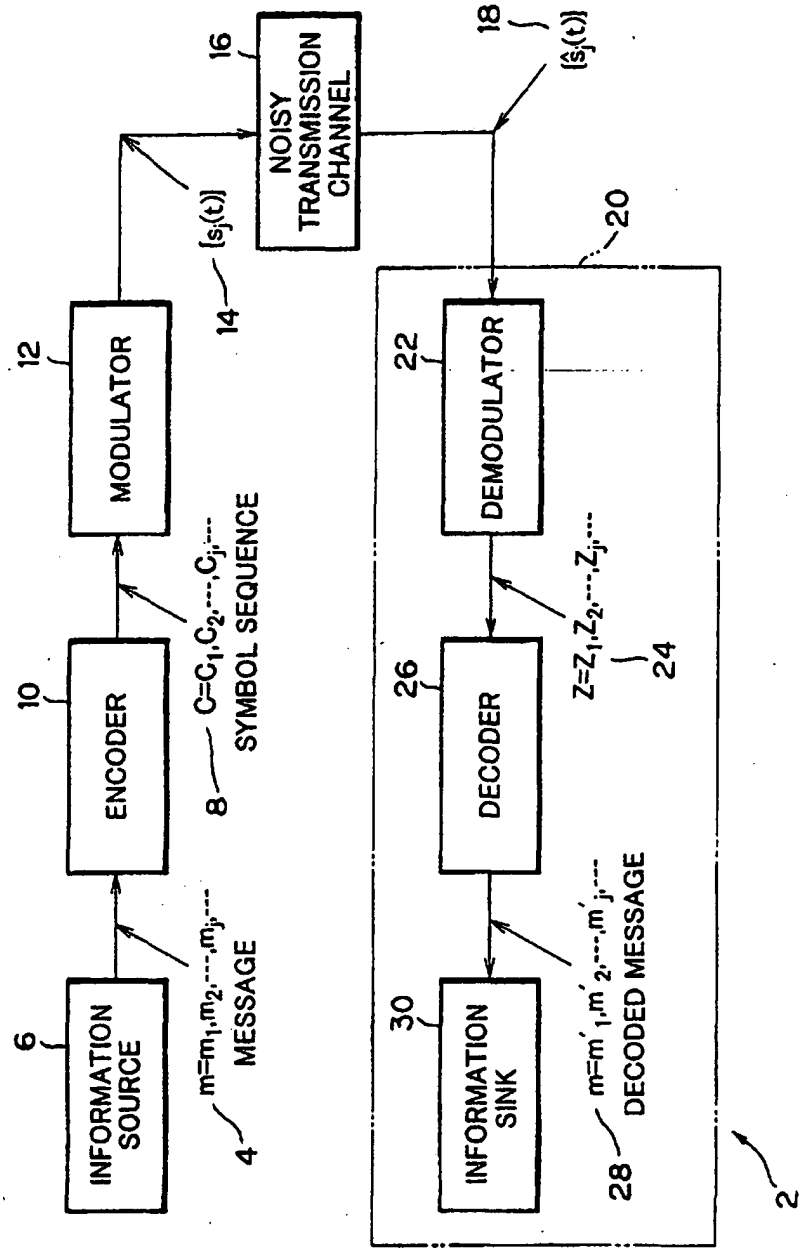

INFORMATION SOURCE
6
m=m1,m2,---,mj,---
MESSAGE
4
ENCODER
10
C=C1,C2,---,Cj,---
SYMBOL SEQUENCE
8
MODULATOR
12
{sj(t)}
14
NOISY TRANSMISSION CHANNEL
16
{ŝj(t)}
18
20
DEMODULATOR
22
Z=Z1,Z2,---,Zj,---
24
DECODER
26
m=m'1,m'2,---,m'j,---
DECODED MESSAGE
28
INFORMATION SINK
30
2

FIG.2

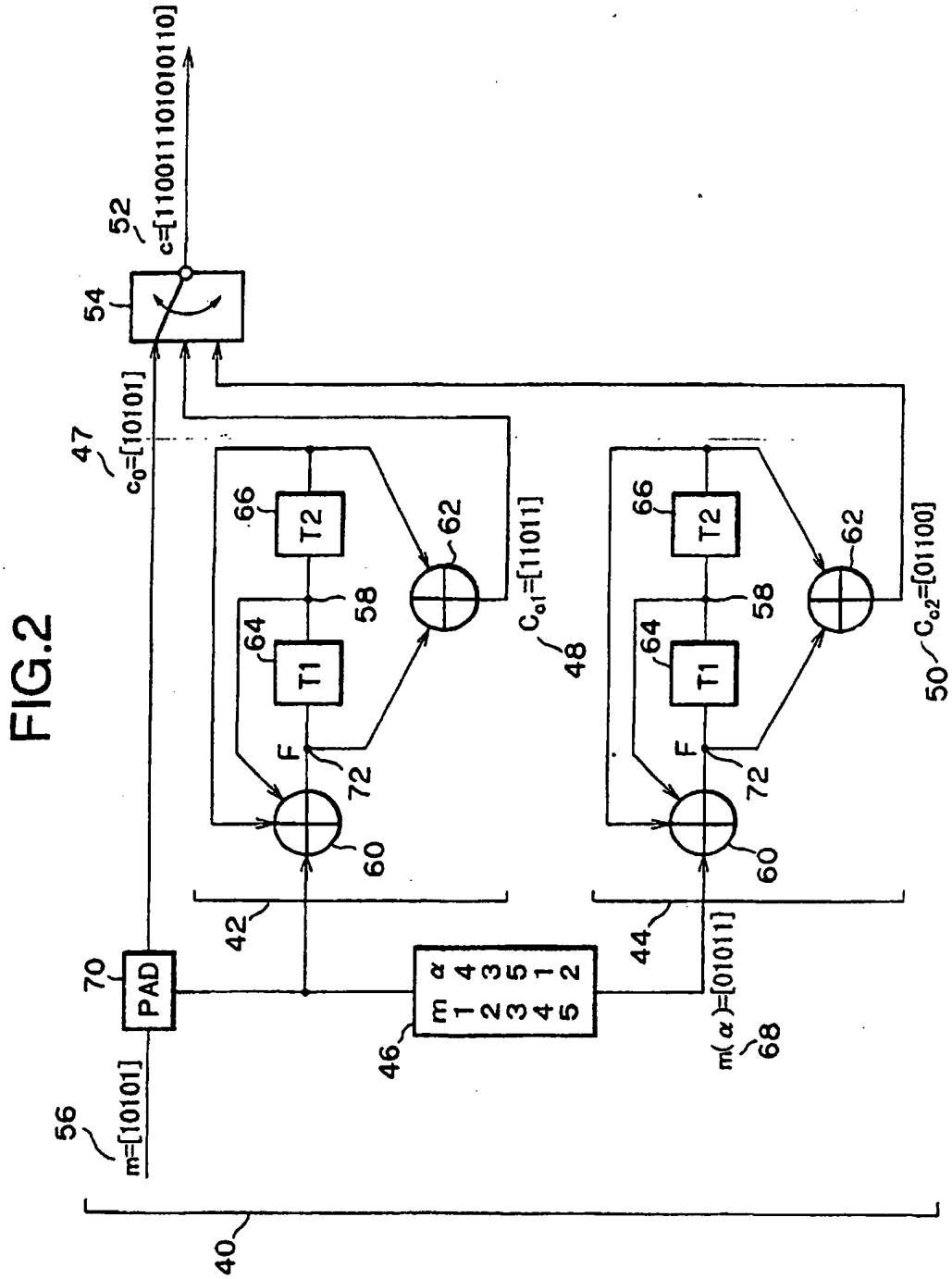

c=[11001110101010110]
52
54
47
c_0=[10101]
66
T2
62
58
64
T1
F
72
60
C_{o1}=[11011]
48
C_{o2}=[01100]
50
42
44
70
PAD
m α
1 4
2 3
3 5
4 1
5 2
46
m(α)=[01011]
68
56
m=[10101]
40

FIG.3

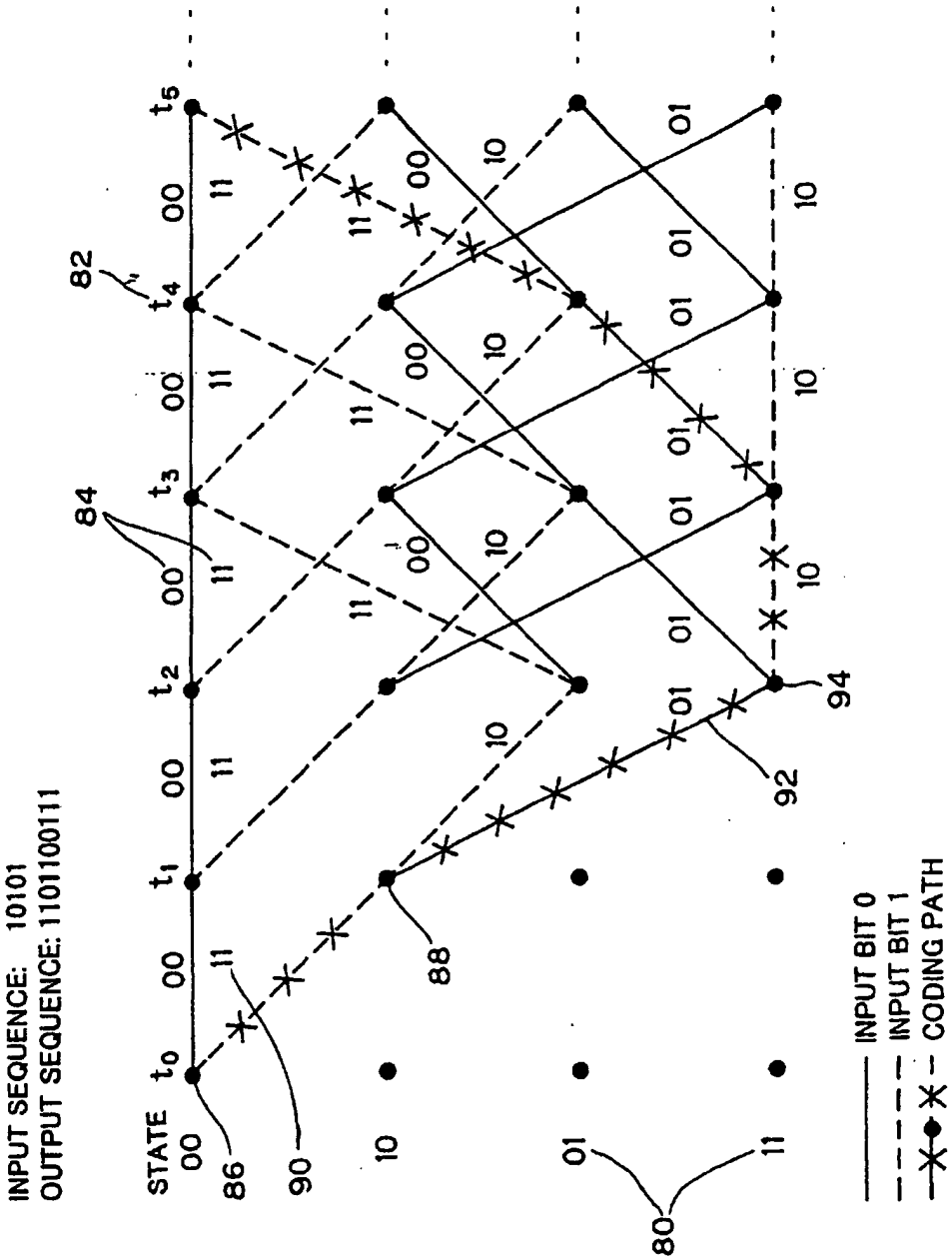

INPUT SEQUENCE: 10101
OUTPUT SEQUENCE: 1101100111
STATE
00
10
01
11
t_0
t_1
t_2
t_3
t_4
t_5
80
82
84
86
88
90
92
94
INPUT BIT 0
INPUT BIT 1
CODING PATH

FIG.4

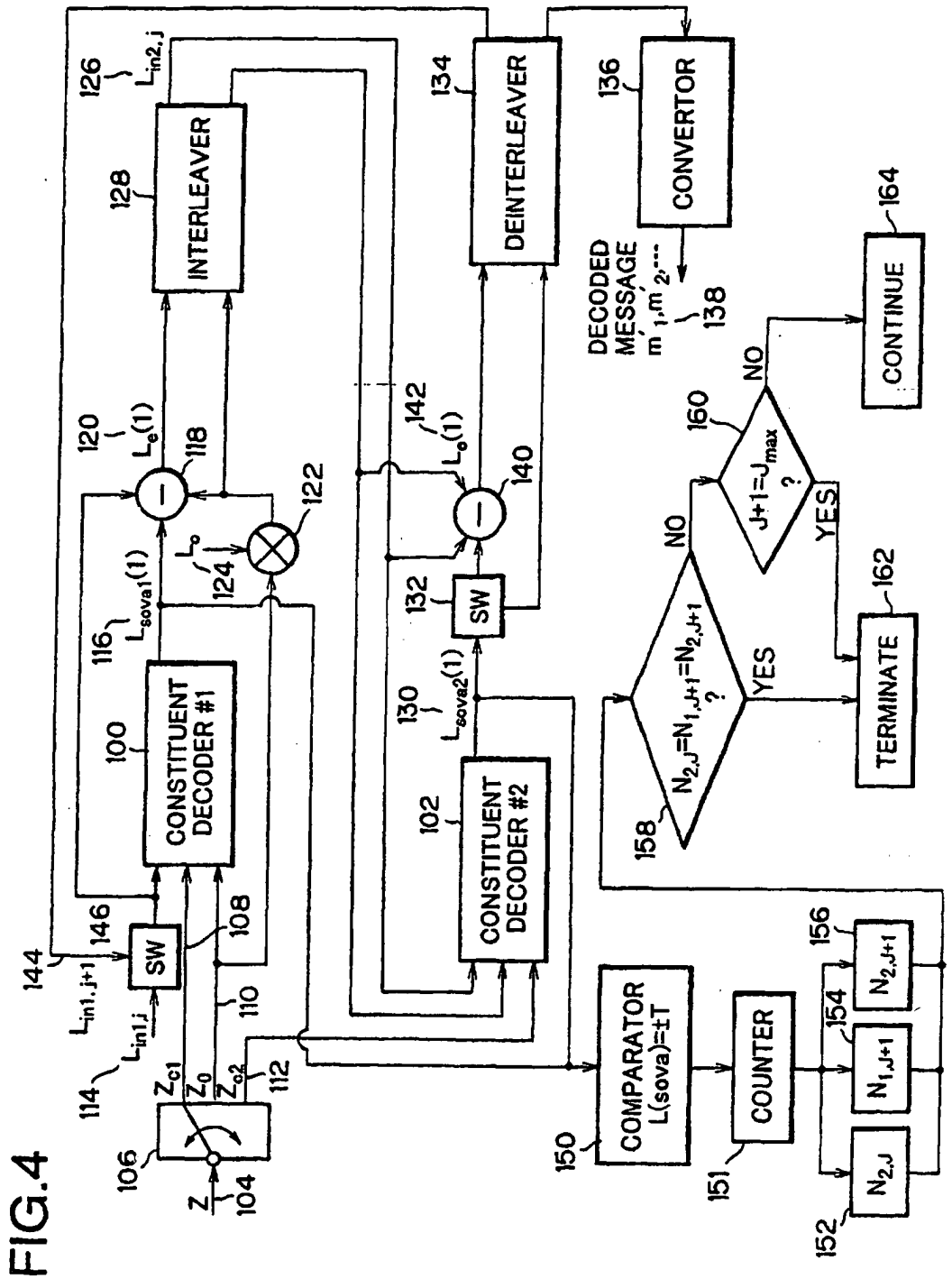
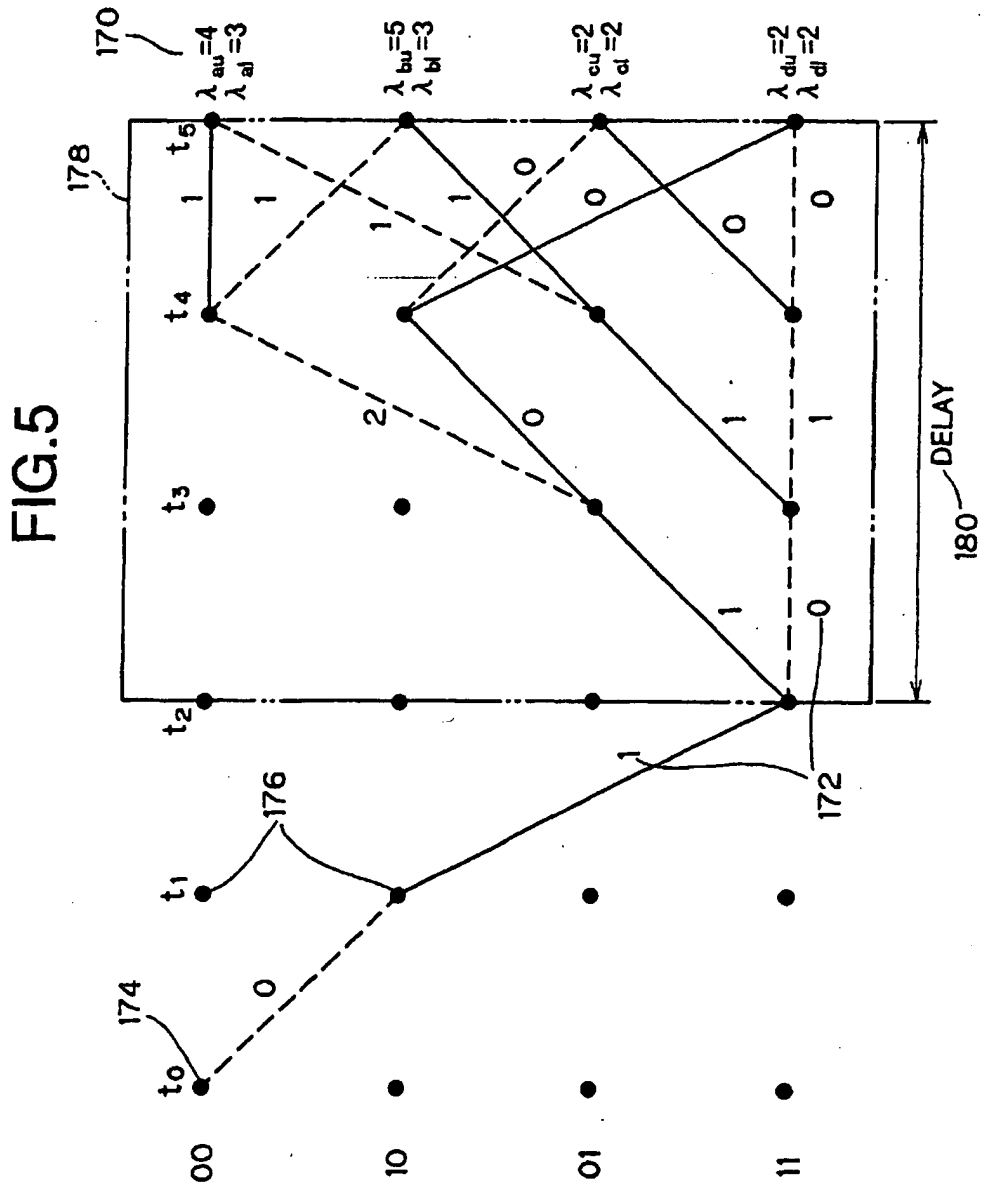

CONSTITUENT DECODER #1
CONSTITUENT DECODER #2
INTERLEAVER
DEINTERLEAVER
CONVERTOR
DECODED MESSAGE
CONTINUE
TERMINATE
COMPARATOR L(sova)=±T
COUNTER
SW
J+1=Jmax ?
Nz,J=N1,J+1=Nz,J+1 ?
NO
YES

FIG.5


170
λ_{au}=4
λ_{al}=3
λ_{bu}=5
λ_{bl}=3
λ_{cu}=2
λ_{cl}=2
λ_{du}=2
λ_{dl}=2
178
t_5
t_4
t_3
t_2
t_1
t_0
DELAY
180
172
176
174
00
10
01
11